

WHAT IS CLAIMED IS:

1 1. A method of debugging code that executes in a
2 multithreaded processor having a plurality of microengines
3 comprises:

4 receiving a program instruction and an identification
5 representing a selected one of the plurality of microengines
6 from a remote user interface connected to the processor

7 pausing program execution in the threads executing in the
8 selected microengine;

9 inserting a breakpoint after a program instruction in the
10 selected microengine that matches the program instruction
11 received from the remote user interface;

12 resuming program execution in the selected microengine;
13 executing a breakpoint routine if program execution in
14 the selected microengine encounters the breakpoint; and
15 resuming program execution in the microengine.

16 2. The method of claim 1 wherein pausing comprises disabling
17 a processor enable bit associated with the selected
18 microengine.

19 3. The method of claim 1 wherein pausing comprises:

20 determining when a context swap between the threads
21 occurs in the selected microengine; and

22 disabling a processor enable bit associated with the
23 selected microengine in response to the context swap.

1 4. The method of claim 1 wherein executing comprises:
2 sending an interrupt to a controlling processor register;
3 and
4 processing the interrupt.

1 5. The method of claim 4 wherein processing comprises:
2 sending the identification to an interrupt handler; and
3 executing the breakpoint routine in the microengine
4 represented by the identification.

1 6. The method of claim 1 wherein the breakpoint routine
2 comprises:
3 writing data to a register.

1 7. The method of claim 6 wherein the data are representative
2 of the state of the threads in the selected microengine.

1 8. The method of claim 4 wherein the controlling processor
2 register comprises a 32-bit register.

1 9. The method of claim 8 wherein bits 6:0 of the 32-bit
2 register represent thread numbers corresponding to the threads
3 in the selected microengine.

1 10. The method of claim 8 where bits 9:7 of the 32-bit
2 register represent whether the interrupt is a breakpoint
3 interrupt.

1 11. The method of claim 1 wherein the breakpoint routine
2 resides in a store of a controlling processor.

1 12. A processor that can execute multiple contexts and that
2 comprises:

3 a register stack;
4 a program counter for each executing context;
5 an arithmetic logic unit coupled to the register stack
6 and a program control store that stores a breakpoint command
7 that causes the processor to:
8 perform a breakpoint routine residing in a debug
9 library in one of the contexts; and
10 resume program execution.

11 13. The processor of claim 12 wherein a breakpoint in the
12 context points to the breakpoint routine.

13 14. The processor of claim 13 wherein the breakpoint is
14 inserted into the context in response to a user request
15 received through a remote user interface connected to the
16 processor.

17 15. The processor of claim 13 wherein an end of the
18 breakpoint routine points to a program counter of the context.

19 16. The processor of claim 12 wherein the breakpoint routine
20 performs at a context switch.

21 17. The processor of claim 13 wherein the breakpoint causes
22 an interrupt.

23 18. The processor of claim 17 wherein an interrupt handler
24 services the interrupt.

1 19. The processor of claim 18 wherein the interrupt handler
2 identifies the context from the interrupt.

1 20. The processor of claim 18 wherein the interrupt handler
2 identifies a processor identification.

1 21. A computer program product, disposed on a computer
2 readable medium, the product including instructions for
3 causing a multithreaded processor having a plurality of
4 microengines to:

5 receive a program instruction and an identification
6 representing a selected one of the plurality of microengines
7 from a remote user interface connected to the processor;

8 pause program execution in the threads executing in the
9 selected microengine;

10 insert a breakpoint after a program instruction in the
11 selected microengine that matches the program instruction
12 received from the remote user interface;

13 resume program execution in the selected microengine;

14 execute a breakpoint routine if program execution in the
15 selected microengine encounters the breakpoint; and

16 resume program execution in the microengine.